REMARKS

The Examiner has objected to the drawings. Such objections are deemed overcome with the submission herewith of formalized drawings.

The Examiner has rejected each of the claims under 35 U.S.C. 103(a) as being unpatentable over Yann et al. (U.S. Application Publication No. 2002/0078368) in view of Risk. (U.S. Patent No. 5,471,629) in further view of Nachenberg (U.S. Patent No. 6,357,008). Applicant respectfully disagrees with such rejection, especially in view of the amendments made hereinabove to each of the independent claims. Specifically, applicant has incorporated the subject matter of dependent Claims 9 and 11 et al. into each of the independent claims.

In response to applicant's latest arguments and amendments, the Examiner has relied on the following excerpts to meet applicant's claimed analysis logic that includes "a dependence table indicating dependency between state variables within said computer and loaded variable values, and for each program instruction said analysis logic makes a determination as to which state variables are read and written by that program instruction and for each loaded variable value within said dependence table if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared" (see this or similar, but not identical language in each of the independent claims).

> "[0030] The present disclosure provides methods of detecting a polymorphic viral code at earlier stages of emulation. The operand and operator values of instructions emulated may be monitored. Special consideration may be given to unused values and those instances in which the operand/operator value is calculated without being defined. A plurality of these instances found during emulation may contribute to a determination that a polymorphic viral code is present. Once the suspect instances are identified, there is a higher probability that the code is viral or used by a virus decryptor. The emulator then is allowed to emulate a larger amount of instructions. Thus, emulation time is spent only on emulating code which has a high probability of containing a virus. ...

> [0034] The conditions of the register/flag triggering an improper register usage state are shown in the state diagram in FIG. 3. During emulation of the program, a register/flag may be in one of the following three states: (a) undefined; (b) set; and (c) used. If a register/flag is loaded with an unknown value, its state is "undefined." Once the value that is loaded into a register/flag is known, the register/flag is in a "set" state. If a value in the register/flag is employed by a CPU instruction, the register/flag is in a "used" state." (see Yann)
>
> "A single Function Dependence table is used to correlate extensional functions with intensional functions. As previously indicated, an intensional function accesses an attribute value by computation or the like based on other attribute values. A change in one of these other values may result in a change in the value accessed by that intensional function. Accordingly, if an attribute which is accessed by an intensional function is being monitored, any change in any of the values which are used in accessing the monitored attribute must be detected in order to determine whether the monitored value has changed. The Function Dependence table provides this correlation.
>
> Accordingly, during monitor definition, if the function which accesses the attribute to be monitored is an intensional function, that function is listed in the Function Dependence table together with each extensional function which could change any of the values which are utilized by that intensional function in accessing the monitored attribute. Later, when Check Monitors is called, the extensional function update calls as listed in the Function Change table are compared with the listings in the Function Dependence table. In this way, the system detects update function calls which may have resulted in changes to monitored attributes (block 302)." (see col. 10, lines 28-51 from Risch)

Further, the Examiner argues that "Yann ... discloses the limitations of part c with the exception that Yann does not disclose the use of storing the data in a table. ... Risch discloses the use of a dependency table in which parameters and associated values are monitored."

In response, applicant respectfully asserts that the Examiner has not taken into consideration the full weight of applicant's claims. Specifically, applicant does not simply claim a dependency table in which parameters and associated values are monitored, but rather a dependence table indicating dependency between state variables within said computer and loaded variable values, such that a determination is made as to which state variables are read and written by that program instruction. More importantly, for each loaded variable value within the dependence table, a specific conditional

operation is carried out. In particular, if any state variable read by that program instruction is marked as dependent upon the loaded variable value, then all state variables written by that program instruction are marked as dependent upon the loaded variable value with previous dependencies being cleared. Both Yann and Risch simply fail to meet these specific emphasized limitations. Only applicant teaches and claims such specific data structure in association with the dependency table and corresponding functionality.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

Applicant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art references, when combined, fail to teach or suggest all of the claim limitations, as noted above. Nevertheless, despite such paramount deficiencies and in the spirit of expediting the prosecution of the present application, applicant has included the following claim language from dependent Claims 9 and 11 et al. into each of the independent claims:

```
"wherein a state variable is marked as initialised upon
occurrence of  one of:
(i) a write to said state variable of a determined initialised
value; and
(ii) use of said state variable as a memory address value by a
program instruction;
wherein a branch path stops being followed when  one of the
following occurs:
(i) there are no further suitable program instruction for
execution within that branch path; and
(ii) said branch path rejoins a previously parsed execution path"
```

(see former dependent Claims 9 and 11 et al., presently incorporated in to each of the independent claims).

With respect to the subject matter of former Claim 9 (now incorporated into each of the independent claims), the Examiner relies on paragraph 34 from Yann (see above) to meet such limitations. However, such excerpts simply disclose setting a flag as "undefined," "set," and "used," based on "an unknown value," "a known [value]," and "a value [that]... is employed by a CPU instruction." None of these meets applicant's claimed state variable that is marked as initialised upon occurrence of one of: a write to said state variable of a determined initialised value or use of said state variable as a memory address value by a program instruction, as claimed.

With respect to the subject matter of former Claim 11 (now incorporated into each of the independent claims), the Examiner relies on col. 10, lines 26-36 below from Nachenberg to meet such limitations.

"The above section of instructions checks to see if the potential host program that it has located (the target host program) is greater than a minimum size (1000h bytes). If the potential host program is bigger, then it will be considered for infection (i.e. the jump is not taken). Otherwise, the potential host program is not considered for infection and the virus merely halts (i.e. the jump is taken). Branch points such as this one are critical because if the virus does not have its criteria for a host program satisfied, the infectious code may remain in an untaken branch which is not explored by a prior art dynamic heuristic antivirus program." (col. 10, lines 26-36 from Nachenberg)

However, such above excerpts merely disclose termination based on a predetermined size. Such predetermined size criteria simply does not meet applicant's claimed condition that "there are no further suitable program instruction for execution within that branch path" or the "branch path rejoins a previously parsed execution path," as claimed. A predetermined size criteria simply does not meet such limitations.

Thus, a notice of allowance or a specific prior art showing of all of applicant's claim limitations, in combination with the remaining claim elements, is respectfully requested.

Thus, all of the independent claims are deemed allowable. Moreover, the remaining dependent claims are further deemed allowable, in view of their dependence on such independent claims.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 505-5100. The Commissioner is authorized to charge any additional fees or credit any overpayment to Deposit Account No. 50-1351 (Order No. NAI1P460/01.045.01).

Respectfully submitted,
Zilka-Kotab, PC.

Kevin J. Zilka
Registration No. 41,429

P.O. Box 721120
San Jose, CA 95172-1120
408-505-5100